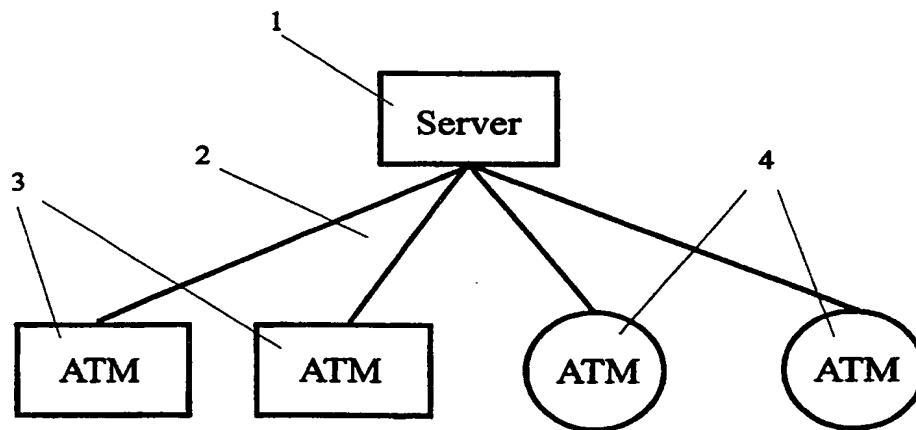




INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G07F 19/00, G06F 17/60		A2	(11) International Publication Number: WO 99/49431
			(43) International Publication Date: 30 September 1999 (30.09.99)
(21) International Application Number: PCT/GB99/00927 (22) International Filing Date: 24 March 1999 (24.03.99) (30) Priority Data: 9806843.0 24 March 1998 (24.03.98) GB (71) Applicant (for all designated States except US): KORALA ASSOCIATES LIMITED [GB/GB]; John Cotton Building, Sunnyside, Edinburgh EH7 5RA (GB). (72) Inventor; and (75) Inventor/Applicant (for US only): KORALA, Aravinda [FR/GB]; KAL, John Cotton Building, Sunnyside, Edinburgh EH7 5RA (GB). (74) Agent: KENNEDY & CO.; Station House, 34 St. Enoch Square, Glasgow G1 4DF (GB).		(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG). Published <i>Without international search report and to be republished upon receipt of that report.</i>	

(54) Title: APPARATUS AND METHOD FOR PROVIDING TRANSACTION SERVICES



(57) Abstract

Apparatus and method for providing transaction services, in particular a computer-based transaction machine, such as an ATM, and a method for providing transaction services using said transaction machine. One or more software applications interact with middleware software through functional interfaces that are hardware independent but provide functionality which is implemented in a manner adapted to the capabilities of the particular hardware implementation. Objects provided for standard transaction functions are independent of the interface between the user and the transaction machine, said interface being customisable. The resulting transaction machines are typically combined into networks and these networks may readily be combined to form an Extranet.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakhstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

1 APPARATUS AND METHOD FOR PROVIDING TRANSACTION SERVICES.

2

3 The present invention relates to apparatus and a method
4 for providing transaction services. In particular it
5 relates to networked computer-based transaction machines
6 and a method for providing transaction services using
7 said transaction machines.

8

9 Transaction machines are herein defined as any computer-
10 based machine able to interact with a user.

11

12 The term ATM is used herein to refer to any transaction
13 machine able to dispense cash. Typically, such machines
14 can also undertake physical transactions such as
15 inputting information through a keypad or touch screen,
16 making sounds, producing video and printing. They might
17 also be able to read bank cards and such like. Kiosks
18 are transaction machines unable to dispense cash, but
19 otherwise able to provide a range of interactive
20 features, often relating to financial services. For test
21 purposes, a conventional PC may be used as a transaction
22 machine.

1

2 Electronic cash machines are a large and rapidly growing
3 market. Many different hardware providers produce
4 equipment for this market such as the machines
5 themselves, the servers to which they connect and the
6 networking means through which they typically
7 communicate. Furthermore, many different operating
8 systems and applications are used both for operating and
9 developing these systems.

10

11 As a result of the complexity and diversity of hardware
12 and software currently being used in this field, it is
13 difficult and expensive to alter these systems to extend
14 their functionality, upgrade to newer and better
15 hardware, software or networking means or to interface
16 with other systems. As it is difficult to make even
17 small changes to complex systems without running the risk
18 of their malfunctioning, the evolution of such systems is
19 slow.

20

21 It would therefore be advantageous to find a way of
22 making it easier to alter the hardware, software and
23 network components of ATMs/kiosks, their servers and
24 their networking means.

25

26 Furthermore, it would be advantageous to provide a means
27 for enabling such changes to be implemented in small
28 stages.

29

30 Yet further, it would be advantageous to find a way to
31 reduce the risk of such systems malfunctioning.

32

1 In current practice, it is difficult and therefore
2 expensive to operate ATM/kiosk networks containing
3 diverse hardware, software and networking means. Often
4 large amounts of hardware and software must be upgraded
5 concomitantly to reduce interface problems. Furthermore,
6 it is difficult to interface networks of dissimilar
7 devices, perhaps belonging to different organisations.
8 If dissimilar ATM/kiosk systems could be readily
9 interfaced, forming a so-called Extranet, new and useful
10 co-operative applications could be developed which,
11 although currently possible, are prohibitively complex
12 and expensive at the present time.

13
14 It would therefore be advantageous to provide a better
15 means of networking ATMs/kiosks which use diverse
16 hardware, software and networking implementations. In
17 particular, it would be advantageous to provide a means
18 of allowing co-operation between dissimilar networks.
19 Furthermore, it would be advantageous to reduce the
20 amount of work required to enable ATM/kiosk applications
21 to run on dissimilar hardware implementations.

22
23 At the present time, there is a rapid growth in
24 electronic commerce (e-commerce), usually conducted over
25 the internet. E-commerce is being limited by
26 difficulties gaining access to the internet for many
27 consumers and due to the limitations of the machines
28 currently used by consumers for internet transactions. A
29 typical e-commerce consumer will access a web site using
30 a home PC. However, home PCs lack facilities such as the
31 ability to dispense cash or read a smartcard which are
32 important in many types of common financial transaction.

1

2 It would therefore be desirable to provide a means of
3 allowing internet-based e-commerce to be accessed from
4 ATMs and kiosks which already have hardware facilities
5 suitable for financial transactions. This would allow e-
6 commerce services to be provided which required expensive
7 or high-security hardware facilities which cannot be
8 securely provided at a reasonable cost on privately owned
9 web browsers. Furthermore, it would be possible for e-
10 commerce to be made readily available to a much larger
11 base of consumers than is currently available.

12

13 The design of ATM networks typically involves input from
14 numerous professionals such as software and hardware
15 engineers specialising in the various systems,
16 applications and communications means, graphics and GUI
17 specialists, language specialists and so forth. In
18 current working practice these specialists are highly
19 dependent on each other and much time and money is spent
20 communicating different requirements amongst people
21 working on diverse areas of a project.

22

23 It would therefore be advantageous to provide a means by
24 which the different specialists working on a project may
25 work more independently. In particular, it would be
26 highly advantageous to provide a means by which the
27 different specialists may customise elements of the
28 application pertaining to their own specialisation
29 without affecting other elements of the application. It
30 would be particularly advantageous if the different
31 specialists were able to use well known prior art
32 authoring tools to prepare aspects of the application.

1 According to the present invention there is provided a
2 method for providing transaction services wherein

3

4 (a) the user of the transaction services interacts
5 with a computer-based transaction machine which is
6 controlled by one or more software applications;

7

8 (b) the software applications interact with the
9 functional interfaces of middleware software, which
10 extends the functionality of an underlying operating
11 system; and

12

13 (c) said functional interfaces provide functionality
14 which is implemented in a manner adapted to the
15 particular hardware capabilities of the transaction
16 machine.

17

18 The computer-based transaction machine may be selected
19 from a group which comprises automatic teller machines,
20 kiosks, electronic point of sale machines and the like.

21

22 Preferably, the middleware software comprises a series of
23 transaction objects and controls for standard device
24 functions.

25

26 More preferably, transaction objects are independent of
27 the interface between the user and the transaction
28 machine; the interface between the user and the
29 transaction machine being customisable.

30

31 Preferably, the controls implement a capabilities
32 interface.

1

2 More preferably, the capabilities interface is able to
3 communicate the capabilities of the control software.

4

5 The applications, objects and controls may be fully
6 concurrent and asynchronous.

7

8 The controls may have a mode in which events are queued
9 up and delivered to the application on demand.

10

11 Preferably, controls can run on the transaction machine
12 even when supported hardware devices are not present.

13

14 More preferably, the middleware software uses one or more
15 open standards for interacting with different hardware
16 systems.

17

18 Preferably, the middleware software only provides
19 cancellation commands for functions which can be
20 successfully cancelled.

21

22 The middleware software may only requires a timeout
23 command to be supplied when it is meaningful to do so.

24

25 Preferably, all controls are persistent.

26

27 More preferably, there is provided a control containing a
28 persistent object.

29

30 Preferably, all errors and transgressions are asserted by
31 the middleware software.

32

1 Preferably, the middleware software provides a trace
2 facility that is always enabled and which logs trace
3 events.

4

5 The middleware software may use a ring buffer to store a
6 log of trace events.

7

8 Preferably, the middleware software writes trace data to
9 memory and then copies it to disk only when the
10 transaction machine is idle.

11

12 Preferably, one or more software applications are hosted
13 in a web browser.

14

15 More preferably, the use of a web browser provides
16 support for software distribution and network
17 connections.

18

19 An additional browser frame may be provided which
20 contains the device controls required to detect events
21 which must be dealt with immediately they occur.

22

23 The middleware software may comprise a series of COM
24 components with a scriptable ActiveX[®] interface.

25

26 The middleware software may comprise a series of
27 Javabeans[™] components with a scriptable interface.

28

29 The use of a web browser may allow conventional web sites
30 to be displayed by the computer-based transaction
31 machine.

32

1 Preferably, the middleware software allows or disallows
2 access to particular web sites according to a rule
3 database.
4

5 The middleware software may be adapted to customise time-
6 out of the display of individual internet web sites.
7

8 Preferably, said computer-based transaction machine is
9 adapted to allow the software applications and middleware
10 to be altered across a network by an authority.
11

12 More preferably, the transaction machine communicates
13 information about its status to a remote monitoring
14 station across a network.
15

16 According to a second aspect of the present invention,
17 there is provided a computer-based transaction machine;
18 wherein said computer-based transaction machine is
19 provided with hardware devices for interaction with users
20 and the exchange of transaction-related information with
21 other machines; wherein said computer-based transaction
22 machine is controlled by one or more software
23 applications; wherein said software applications control
24 hardware devices through functional interfaces with
25 middleware software; wherein said middleware software
26 extends the functionality of an underlying operating
27 system and wherein said functional interfaces are
28 hardware independent but provide functionality which is
29 implemented in a manner adapted to the capabilities of
30 the particular hardware devices which are provided.
31

1 The computer-based transaction machine may be selected
2 from a group which comprises automatic teller machines,
3 kiosks, electronic point of sale machines and the like.

4

5 Preferably, the middleware software comprises a series of
6 transaction objects and controls for standard device
7 functions.

8

9 More preferably, transaction objects are independent of
10 the interface between the user and the transaction
11 machine; the interface between the user and the
12 transaction machine being customisable.

13

14 Preferably, the controls implement a capabilities
15 interface.

16

17 More preferably, the capabilities interface is able to
18 communicate the capabilities of the control software.

19

20 The applications, objects and controls may be fully
21 concurrent and asynchronous.

22

23 The controls may have a mode in which events are queued
24 up and delivered to the application on demand.

25

26 Preferably, controls can run on a transaction machine
27 even when supported hardware devices are not present.

28

29 More preferably, the middleware software uses one or more
30 open standards for interacting with different hardware
31 systems.

32

1 Preferably, the middleware software only provides
2 cancellation commands for functions which can be
3 successfully cancelled.
4

5 The middleware software may only requires a timeout
6 command to be supplied when it is meaningful to do so.
7

8 Preferably, all controls are persistent.
9

10 More preferably, there is provided a control containing a
11 persistent object.
12

13 Preferably, all errors and transgressions are asserted by
14 the middleware software.
15

16 Preferably, the middleware software provides a trace
17 facility that is always enabled and which logs trace
18 events.
19

20 The middleware software may use a ring buffer to store a
21 log of trace events.
22

23 Preferably, the middleware software writes trace data to
24 memory and then copies it to disk only when the
25 transaction machine is idle.
26

27 Preferably, one or more software applications are hosted
28 in a web browser.
29

30 More preferably, the use of a web browser provides
31 support for software distribution and network
32 connections.

1

2 An additional browser frame may be provided which
3 contains the device controls required to detect events
4 which must be dealt with immediately they occur.

5

6 The middleware software may comprise a series of COM
7 components with a scriptable ActiveX[®] interface.

8

9 The middleware software may comprise a series of
10 Javabeans[™] components with a scriptable interface.

11

12 The use of a web browser may allow conventional web sites
13 to be displayed by the computer-based transaction
14 machine.

15

16 Preferably, the middleware software allows or disallows
17 access to particular web sites according to a rule
18 database.

19

20 The middleware software may be adapted to customise time-
21 out of the display of individual internet web sites.

22

23 Preferably, the computer-based transaction machine is
24 adapted to allow the software applications and middleware
25 to be altered across a network by an authority.

26

27 More preferably, the transaction machine can communicate
28 information about their status to a remote monitoring
29 station across a network.

30

31 According to a third aspect of the present invention
32 there is provided a network comprising a plurality of

1 computer-based transaction machines, one or more
2 networking means and one or more application servers.

3

4 According to a fourth aspect of the present invention,
5 there is provided an Extranet formed by combining a
6 plurality of networks of computer-based transaction
7 machines.

8

9 Preferably, the Extranet is provided with a security
10 mechanism which limits the hardware functionality
11 available to individual software applications.

1 An example embodiment of the present invention, referred
2 to as the system, will now be described with reference to
3 the following Figures wherein:

4

5 Figure 1 shows a simple ATM network;

6 Figure 2 shows an ATM network with diverse hardware;

7 Figure 3 shows two distinct networks being combined
8 to form an Extranet; and

9 Figure 4 shows the software architecture of the
10 preferred implementation of the system.

11

12 Figure 1 shows a simple ATM network comprising a server
13 1, a networking means 2 and an ATM 3. The system is
14 designed to operate such networks and also more complex
15 networks such as shown in Figure 2 wherein there may be
16 ATMs of different functionality, here labelled 4.

17

18 A particular benefit of the system is its ability to
19 allow distinct networks to operate together as shown in
20 Figure 3. Here, two distinct networks 5 and 6 operated
21 by distinct servers 7 and 8 are connected 9. The
22 resulting joined network is referred to as an Extranet.

23

24 By joining multiple networks together, it becomes
25 possible for different organisations to co-operate in the
26 provision of ATM/kiosk network services. For example,
27 suppose that a bank which owned a series of conventional
28 ATMs and an airline which owned a series of ticketing
29 kiosks chose to co-operate. There exists the potential
30 for the bank's ATMs to both allow customers to pay for an
31 airline ticket and to print out that ticket. Similarly,
32 the airline might offer a limited selection of banking

1 services, such as balance display, which are compatible
2 with the functionality of their kiosks.

3
4 Using prior art, the development of such a system would
5 be complex, particularly due to the different hardware
6 and capabilities of the bank's ATMs and the airline's
7 kiosks. Such co-operation between organisations is by no
8 means impossible at the present time, but is currently
9 rare due to the complexity and expense required for
10 implementation.

11
12 In general, the system provides a means for a plurality
13 of servers to operate a plurality of ATMs and kiosks
14 using a plurality of networking means. An example
15 application would be to allow consumers to purchase eg
16 cinema, theatre and airline tickets from different
17 organisations through ATMs positioned at convenient
18 locations.

19
20 Typically, the networking means will be the internet, a
21 corporate intranet or LAN but may be any networking means
22 or a mixture of networking means.

23
24 The system comprises a middleware software layer which
25 extends the function of an underlying operating system
26 and which in turn provides a single programming interface
27 for an ATM/kiosk control application to be written to.

28
29 Figure 4 shows the software architecture of the preferred
30 implementation of the system. An ATM/kiosk control
31 application 10 is hosted in a web browser 11 such as
32 Microsoft's Internet Explorer. The application runs on a

1 computer with a particular operating system, 12, such as
2 Windows NT*, the functionality of which has been extended
3 by middleware software 13.

4
5 The middleware comprises a series of components and
6 objects, for use by the application, which extend the
7 functionality of the operating system and provide tools
8 to simplify development of the ATM application.

9
10 In the preferred implementation all of the system's sub-
11 systems are implemented as a series of COM components
12 with an ActiveX* interface or as Javabeans™ with a
13 scriptable interface. This architecture enables
14 applications running within Internet Explorer to access
15 functionality provided by the operating system and the
16 middleware, including access to hardware.

17
18 A useful benefit of this implementation is that
19 applications may be prepared using common authoring tools
20 and such as Microsoft*'s FrontPage*, VisualStudio*, Visual
21 Interdev* and common development environments such as
22 Visual Basic*, Visual C++*, Powerbuilder*, Delphi* etc.
23 This means that applications can be prepared with tools
24 with which developers will be familiar and which, due to
25 their popularity, provide facilities and support that
26 would be prohibitively expensive to prepare for a custom
27 development environment.

28
29 A further benefit of using browser technology is that
30 they provide an environment in which software download
31 can be readily controlled. The application may be held
32 entirely locally to an ATM/kiosk, entirely on a server or

1 any compromise between these two extremes. The
2 application can be downloaded daily if required.

3

4 The system uses the Windows[®] Open System Architecture
5 Extensions for Financial Services (WOSA XFS) to support
6 ATM hardware in a vendor independent manner.

7

8 The system also uses the Object Linking and Embedding for
9 Point Of Sale (OPOS) standard for interacting with
10 different hardware systems. This means that applications
11 can access hardware independent of whether the underlying
12 hardware supports WOSA XFS or OPOS.

13

14 The system also supports the PC/SC standard for
15 smartcards, thereby providing a uniform way of accessing
16 smartcards.

17

18 Furthermore, the system also provides support for a
19 variety of other open standards such as OFX and SNMP and
20 transaction monitors such as NCR's TOPEND[®].

21

22 Clearly, support for additional standards may readily be
23 added.

24

25 The primary subsystems of the middleware software
26 comprise a series of wizards, device controls, self-
27 service controls, communications controls and status
28 monitoring components.

29

30 The top level components are the wizards, which are a
31 series of transaction objects that implement common
32 ATM/kiosk transactions such as dispensing cash, printing

1 a statement etc. In the preferred embodiment, each is
2 implemented as an ActiveX[®] object or a Javabeans[™]. Whilst
3 wizards are running, they take control of the function of
4 the ATM/kiosk. Wizards interface with other controls and
5 encode all of the top-level control logic.

6
7 Applications can be built with the system by customising
8 and combining wizards. Wizards encapsulate all of the
9 features and functionality required by a particular
10 transaction or chunk of application. When using ActiveX[®].
11 Wizards receive input via ActiveX[®] properties and methods
12 and output their state as a set of ActiveX[®] events.
13 Alternatively the wizard can be implemented in the same
14 way as a Javabeans[™]. As a result of this design feature,
15 the wizard is completely independent of the ATM/kiosk-
16 user interface.

17
18 For example, an ATM might have a single button which
19 dispenses \$10 on demand. A second ATM might implement
20 more complex controls and display a detailed animation
21 whilst money is issued. However, the same wizard may be
22 used to implement both these ATMs. The wizard
23 encapsulates the essential software logic of the
24 transaction while allowing the user interface to be
25 freely defined by script on the browser page.

26
27 This has several important benefits which will lead to
28 time and cost savings: firstly, the encapsulated features
29 within the wizard can be reused between different
30 applications whilst allowing the different applications
31 to have totally different look and feel. Secondly, this
32 allows the user interface to be designed with common web

1 tools. Thirdly, the user interface may be designed
2 without any risk of compromising the function of the
3 wizard. Finally, the user interface may be designed by a
4 specialist who may not be an expert in the other aspects
5 of ATM/kiosk software and hardware.

6
7 An additional important feature of the wizards is that
8 they are able to interpret the capabilities of the
9 hardware on which they are run. For example, they may be
10 able to establish whether a cash dispensing means is
11 available. One application may then run on a plurality
12 of different hardware implementations, adapting its
13 functionality to the capabilities of that hardware.

14
15 This not only allows different hardware implementations
16 to be incorporated into the same network but allows
17 distinct networks to be joined into an Extranet.

18
19 The device controls provide hardware independent access
20 to the special devices on an ATM or kiosk. Each device
21 control acts as a persistent server that can be
22 controlled and interrogated by one or more applications
23 or wizards. A device control abstracts the details of
24 the hardware underneath it and acts as a complete server
25 for that device. Applications and wizards interact with
26 controls through a scriptable ActiveX® interface or a
27 Javabeans™ interface.

28
29 Some example device controls supported by the system are:

- 30 • Camera
31 • Card Reader (motorized, swipe, DIP, smart cards etc.)

- 1 • Cash Acceptor
- 2 • Cash Dispenser
- 3 • Coin Dispenser
- 4 • Depository
- 5 • Doors
- 6 • Encryptor
- 7 • Guide Lights
- 8 • Indicators
- 9 • Journal Printer
- 10 • Keyboards
- 11 • Laser Printers
- 12 • Modems
- 13 • Operator Panel
- 14 • Passbook (including page turn)
- 15 • Pin Pad
- 16 • Receipt Printer
- 17 • Scanner
- 18 • Sensors
- 19 • Signature Capture
- 20 • Statement Printer
- 21 • Touchscreen
- 22 • UPS
- 23 • VendorMode
- 24 • Weighing Scales

25

26 Multiple applications may be run simultaneously and
27 device controls are fully concurrent. This is important
28 as the cycle time of ATMs and kiosk transactions can be
29 critical. Their design is such that they can be used in
30 an event-driven manner, with controls reporting their

1 result (success or failure) via ActiveX® or Javabeans™
2 events, or in a procedural manner from within a language
3 such as C++. In the event-driven mode, applications can
4 be readily created using browser technology; for example,
5 readily available web tools which provide appropriate
6 easy-to-use graphical interfaces can be used to create
7 event-driven applications.

8
9 In order to be able to operate asynchronously, all
10 controls create their own thread, called the event
11 thread, when first constructed. When an asynchronous
12 method is called, a command message is sent to the event
13 thread. The event thread carries out the command and
14 sends a message back to the main thread on completion:
15 the completion method causes the appropriate event to be
16 fired. By implementing commands using the event thread,
17 the main application thread is free to process other
18 tasks in parallel. The event thread also ensures that
19 the device states persist from one application page to
20 another: although controls on browser pages are being
21 continually created and destroyed, the event thread
22 remains running and ensures that the connection to the
23 device is never lost.

24

25 When controls are run in a procedural manner, from a
26 language such as C++, the controls may be set to a mode
27 in which events are queued up and delivered to the
28 application on demand, allowing the application to carry
29 out other tasks, and return to the event queue at an
30 appropriate time.

31

1 The self-service controls provide the functionality
2 necessary for creating self-service applications.
3 Important self-service controls are described further
4 below. The communications controls provide access to the
5 remote host computers. Both the self-service and
6 communications controls have the same server architecture
7 as the device controls and all may be executed
8 asynchronously.

9
10 The status monitoring system monitors the health of the
11 ATM or Kiosk and sends status and alert signals to an
12 external monitoring station using SNMP alerts.

13
14 All controls implement a capabilities interface, allowing
15 an application or wizard to interrogate the capabilities
16 of the control as well as the device which the control
17 represents.

18
19 Therefore, not only can different hardware
20 implementations be integrated into the same network or
21 Extranet, the applications can dynamically configure the
22 services they provide depending on the capabilities of
23 the hardware available on the kiosk.

24
25 As a result of this design, individual software
26 components can be upgraded without having to change other
27 aspects of the application. New features can be added
28 without making the application dependent on those
29 features.

30
31 Furthermore, hardware and networking components may be
32 upgraded or altered step by step. Due to the modular

1 nature of the system and its customisability, a plurality
2 of communications and hardware implementations may be
3 used at once. This means that an organisation which runs
4 an ATM/kiosk network might use its legacy communications
5 and hardware implementations, perhaps concurrently with
6 Internet/Intranet support. This means that ATM networks
7 may be implemented and altered step-wise.

8

9 Such upgrades are particularly easy when using the Open
10 Financial Exchange (OFX) architecture. The middleware
11 software implements a single OFX Control which may
12 interface with an OFX server by any networking means.
13 The OFX server may also interface with a host by any
14 networking means. Once this architecture is implemented,
15 the resulting network topology may be readily altered,
16 making this an easy migration path for existing networks
17 to use this system.

18

19 A further implication of the design of the controls is
20 that they can run on an ATM/kiosk even when actual
21 hardware devices are not present. This allows the
22 applications to be started up and run, for example for
23 development and test purposes, without requiring
24 particular hardware. When the application requests the
25 capabilities of a particular control, the control will
26 reply that the device is not present and that the
27 capabilities are null. Therefore it is possible to
28 create and test application on, for example, a PC. In
29 this situation, the PC will behave like an ATM/kiosk in
30 its interactions with the application.

31

1 An ignore mode is also provided wherein particular
2 controls will return "success" for every command. This
3 allows the application to use generic code which does not
4 need to test whether the device is present at each step,
5 simplifying the code that needs to be written when
6 creating an application to cope with various hardware
7 capabilities.

8
9 An HTML-based application is also provided with the
10 system for testing device controls. This application
11 allows the operator to select a subset of the devices for
12 testing. For each device, two test sequences are
13 defined: one requires operator interaction (e.g.
14 entering/removing a card) and one requires no operator
15 interaction. When the latter is selected, the
16 interaction-free test sequences will be repetitively run
17 for the selected devices, allowing applications provided
18 using this system to be easily stress tested. Complete
19 tests including operator interaction may also be
20 selected. Testing is automated and therefore as
21 reproducible as possible.

22
23 All controls include a security mechanism. This
24 mechanism allows the methods of the various controls to
25 be enabled and disabled. This is particularly important
26 in an Extranet environment when applications of differing
27 abilities run on a given kiosk or ATM. For example, if a
28 bank operating a network of ATMs allowed an airline to
29 dispense tickets through its ATMs by way of an Extranet,
30 it would wish to disallow the airline's application from
31 dispensing cash.

32

1 This security mechanism is implemented by a key passing
2 technique as follows:

3
4 The middleware software contains a security control which
5 allows the current security configuration of an ATM or
6 kiosk to be set. Using the security control, the owner
7 of the ATM or kiosk can specify details of the security
8 configuration (i.e. which methods of a control are
9 allowed and disallowed). Applications identify
10 themselves to the security control via a digital
11 certificate which sets the security configuration as
12 specified by the ATM/kiosk owner. If the application
13 attempts to call a disallowed method of control, a trap
14 is generated, transferring control to the ATM/kiosk
15 owner's application.

16
17 An important benefit of the system is that it may readily
18 be used to provide internet based e-commerce facilities
19 through ATMs and kiosks, not only allowing e-commerce
20 facilities to be used by a larger consumer base but also
21 enabling e-commerce which requires expensive or high-
22 security hardware facilities such as cash dispensers or
23 identity verification means that cannot readily be
24 provided on privately owned PCs and web-browsers.

25
26 To help enable this, the system provides a Site-Minder
27 control which allows existing web sites to be safely
28 delivered via ATMs and kiosks. This control provides
29 several important features. For example, it monitors the
30 URL of each page of the web-site being delivered and
31 allows or disallows the page according to a rules
32 database. This stops the user from straying into other

1 web-sites or web-pages that are not normally part of the
2 purpose of the ATM/kiosk. The control allows each page
3 to be given a customised time-out which is important as
4 web sites are normally designed for use at home and have
5 different (longer) time-outs than would be appropriate
6 for public ATMs/kiosks. Web pages may be navigated using
7 a touch sensitive screen, making them intuitive and easy
8 to use. The control can also magnify small features on a
9 web page (such as hypertext links and images with links)
10 This magnification can be toggled on and off by the user,
11 thereby animating the hypertext link. This is beneficial
12 firstly because it makes it easier for the user to see
13 where the link is and secondly because it becomes easier
14 for the user to select the link when it is in its
15 magnified state.

16

17 An additional feature provided by the system for use with
18 ATMs/kiosks with touchscreens is a "softkeyboard" wherein
19 a keyboard is displayed on the touch screen and contact
20 with the displayed keyboard is interpreted by the system
21 like keystrokes on a real keyboard, thereby removing the
22 need for a physical keyboard to be provided.

23

24 One problem commonly faced by web designers is that
25 objects placed on a web page are destroyed when the page
26 is changed. A useful benefit of the middleware is that
27 the ActiveX[®] hook idea solves this problem - underlying
28 objects remain persistent while lightweight hooks on each
29 page access the object.

30

31 Lack of persistence also leads to problems for the
32 application developer in storing application-wide data.

1 A solution to this problem is provided by a scratchpad
2 control which has a persistent object at its core and
3 allows the application to store and retrieve data at any
4 time. This control supports the Vbscript variant type,
5 allowing all types of data to be stored and retrieved.
6 Furthermore, this control allows data to be shared
7 between multiple applications, marking it as shared.

8
9 A related problem when implementing web-based ATM
10 applications relates to events which must be dealt with
11 immediately, no matter when the event occurs. For
12 instance, if a safe door is opened, an application may
13 need to shut down immediately. This would not be easy to
14 implement in a web-based environment as every page would
15 have to contain some code to handle the event. This
16 problem can be solved in the system by operating a
17 second, invisible frame alongside the main application
18 frame. The invisible frame contains all the device
19 controls needed to detect the events that must be reacted
20 to. This frame may then take control, perhaps closing
21 down the main frame.

22
23 Error handling in traditional ATM applications is
24 difficult. Components may return a large number of error
25 cases, resulting in complex code.

26
27 The middleware software separates the responses it sends
28 to the application into "good responses" and error
29 responses. Most commands have a single good response and
30 all errors are mapped to a single error response,
31 although some may have a plurality of good responses.
32 Good responses allow the application to continue. When

1 an error response is returned, the current transaction
2 flow is normally aborted and control flow jumps out of
3 the normal flow process to handle the error situation.
4 The application can then interrogate the control to
5 determine the exact cause of the error.

6

7 A benefit of this approach is that normal flow is not
8 cluttered by handlers for each of the error cases which
9 can occur. Control may be transferred to generic error
10 handlers which can either recover from the error or abort
11 the transaction completely, perhaps even rebooting the
12 ATM/kiosk. Application code can therefore remain as
13 clear and concise as possible whilst encouraging the
14 application developer to handle all error cases by
15 calling an error handler. In the development
16 environment, fatal errors result in a message box being
17 displayed. A single type of event, DeviceError, is
18 generated when there is some kind of hardware failure,
19 allowing error handling for hardware failure to be
20 encapsulated rather than scattered over many error
21 handlers.

22

23 The system requires applications to interact with it in a
24 well defined way. Even small transgressions are detected
25 and error responses generated; when this happens, the
26 current environment is abandoned and the application is
27 terminated.

28

29 This is based on the well known software engineering
30 approach of assertion; however, the system's assertion
31 differs from common practice by asserting absolutely all
32 disallowed cases, whether serious or not. As a result of

1 this strategy of escalating errors to maximum
2 seriousness, errors are found earlier at development time
3 or at system test time and never allowed to reach a live
4 environment. Although there is a risk of the application
5 reporting a fatal error in the field for a relatively
6 minor problem, this strategy achieves a particularly high
7 level of robustness in comparison to prior art software
8 applications.

9
10 An additional error-handling feature is provided by the
11 way in which the system deals with tracing. In software
12 engineering, tracing is typically enabled only when a
13 problem is suspected; however, this can affect the
14 dynamics of a program, making it harder to find bugs.
15 This is a particularly substantial problem when dealing
16 with time-critical ATM/kiosk applications. However, if
17 conventional tracing was simply always enabled throughout
18 both development and operation of the ATM/kiosk, there
19 would be both performance problems due to, for example,
20 the time spent writing to a hard drive and large quantity
21 of disk space required to store the large number of trace
22 events that will typically be produced.

23
24 The middleware software provides a trace control which
25 records all trace events of the application and
26 underlying middleware and is always enabled. Performance
27 problems are dealt with by writing trace data to memory
28 and writing to disk only when the ATM/kiosk is idle.
29 Cash-dispensing machines and kiosks go through an idle
30 cycle between two users which provides sufficient time to
31 write to disk, even when people are queuing at the
32 machine. Disk space problems are eliminated by using a

1 ring buffer of fixed file size, allocated at boot-up and
2 constant in size throughout operation. When the buffer
3 is full, the oldest data is overwritten, thereby leaving
4 a continual record of the most recent events.

5

6 As a result of this tracing strategy it is much easier to
7 understand one-off or rare problems, which is not easily
8 done when tracing is enabled only once a problem has been
9 reported.

10

11 Furthermore, some ATM/kiosk vendors provide a limited
12 amount of non-volatile RAM. When this is provided, the
13 trace control writes the most recent trace information to
14 this RAM in a ring buffer fashion. As this is very
15 quick, it does not produce any performance problems.
16 However, if the ATM/kiosk freezes up or crashes, the RAM
17 contains the trace of what happened immediately before.

18

19 In addition to the traditional way that ActiveX[®] fires
20 events to the container, the device and self-service
21 controls are able to queue up events and return them one
22 by one when requested. This allows C++ applications to
23 be written in a procedural fashion rather than simply in
24 an event driven fashion. By queuing up these events and
25 delivering them to the application only on demand, the
26 system allows procedural code to be written and makes it
27 easier to develop and maintain the complex logic required
28 in self-service applications.

29

30 Important self-service controls are described below:

31

- 1 • Watchdog control: runs in a separate Windows NT[®]
2 process and reboots the ATM/kiosk if the application
3 crashes. This is achieved by regularly polling the
4 application to check that it is functioning correctly.
5 This control can also be used to daily reboot the
6 ATM/kiosk. The watchdog can monitor multiple
7 applications on a single ATM.
- 8 • System Escape control: used to reboot the ATM/kiosk.
9 Exits in a customisable manner. This control ensures
10 that cached data (eg in the DataCollect control and the
11 Trace control) is flushed to disk before rebooting.
- 12 • DataCollect control: allows application to collect raw
13 data for statistical purposes. It logs and timestamps
14 the various events. As with the Trace control, it logs
15 to memory and then stores on hard disk only when the
16 ATM/kiosk is idle due to the time required to write to
17 the hard disk. Storage by this control is of a fixed
18 size allocated at start-up and remaining constant
19 throughout operation. Storage is in the form of a ring
20 buffer. Typically, the collected data would be
21 exported to a remote location for analysis.
- 22 • Trace control: described above.
- 23 • Scratchpad control: described above.
- 24 • Supervisor application: run simultaneously as a
25 separate application. This means that on an ATM/kiosk
26 with a rear screen, the operator can interact with the
27 ATM/kiosk without taking the machine offline. It
28 allows the operator to access statistics etc. while the
29 machine is still being used. Alternatively, the
30 machine may be taken off-line for intrusive
31 maintenance. In this case, the supervisor application

1 provides an off-line mode with a limited subset of the
2 on-line features.

3 • Security control: described above.

4 • Registry control: allows Windows NT[®] registry to be
5 manipulated by the application.

6 • DirectoryTree control.

7 • Application Launcher control.

8 • INI file control: allows Windows[®] INI files to be read
9 from the browser.

10 • Timed FTP. This allows statistics files and trace files
11 to be sent via the FTP mechanism on a timed basis to an
12 offsite location. (eg daily or weekly).

13 • Key capture control: allows special Windows[®] key
14 combinations such as ctrl-alt-del and alt-tab to be
15 captured where a full PC keyboard is provided.

16 • Popup suppression control. Monitors and captures popup
17 windows originating from the operating system. This
18 makes it easier to allow software components from other
19 vendors to be used in self-service applications. Most
20 third-party software is not intended for self-service
21 applications and expects to be able to interact with
22 the user through popup windows. This is unacceptable
23 in a self-service environment where the main
24 application must have a complete monopoly over the user
25 dialog. This control alleviates this problem by
26 monitoring popups and rapidly executing a pre-
27 determined sequence of tasks, for example hiding the
28 popup and pressing the OK button.

29 • Global config file control. Allows configuration data
30 for ATM networks to be centrally held in a single
31 distributable file. Each ATM/kiosk can query this

- 1 control to retrieve the configuration data which is
2 specific for that ATM/kiosk. This allows variation
3 between individual ATMs/kiosks to be handled in a
4 global way.
- 5 • Telephony control. Allows modems and telephone handsets
6 to be integrated.
 - 7 • SSMS control. Allows software to be downloaded and
8 installed in a controlled manner. This control checks
9 for installation failures and allows the system to
10 recover to a well defined state.
 - 11 • Screensaver control. This control allows the
12 application to jump to a defined web page if the user
13 has been inactive for more than a pre-determined time.
 - 14 • Multiple language control. This control allows the
15 language on a web page to be dynamically modified. It
16 does this by retrieving text strings and graphics from
17 a database on the kiosk. This means that the user may
18 change languages from any browser page - and therefore
19 at any stage of the application.
 - 20 • Clock synch control. This allows the application to
21 synchronize its clock with a server clock, taking into
22 account possible differences in timezone between kiosk
23 and server and taking into account the possibility of
24 large timelags for communication between the kiosk and
25 the server.
- 26 Use of the self-service controls plus additional features
27 of the system and underlying operating system allow
28 ATMs/kiosks to be managed from a remote location. For
29 example, the system supports:
- 30 • Daily software downloads from a remote web server.
 - 31 • Daily reboot and system check.

- 1 • Daily FTP of statistics data to a remote monitoring
2 station.
- 3 • Daily FTP of trace data to a remote monitoring system.
- 4 • Regular health checks of the kiosk (typically every 5
5 minutes).
- 6 • Sending a regular "heartbeat" message to a remote
7 monitoring station. Monitoring of this message allows
8 the fact that the device is continually functioning to
9 be monitored.
- 10 • Allowing direct secure access to the kiosk over a
11 network, for example the Internet, from a remote
12 location.
- 13 • Allowing software maintenance over a network, for
14 example the Internet, from a remote location.
- 15 • Allowing manual reboot of the kiosk over a network, for
16 example the Internet, from a remote location.

17

18 Although hardware is accessed via the WOSA XFS standard,
19 which assigns a different number to each command, the
20 controls have differently named methods and events
21 associated with each operation, making application
22 development easier. WOSA commands may typically generate
23 30-50 events. This wastes time for the application
24 developer and increases the possibilities of error. The
25 middleware reduces the set of possible outcomes to a
26 small number of clearly named completion events, making
27 it easier for the application developer to write reliable
28 code quickly. Outcomes which can only happen if there is
29 a bug in the application cause fatal errors to be
30 triggered.

31

1 The system automatically opens a WOSA XFS session when a
2 device control is first used; there is therefore no need
3 to manually call an Open method. WOSA sessions are
4 maintained between pages through the use of event
5 threads, described above.

6
7 All WOSA XFS methods require a timeout to be provided;
8 however, this is not appropriate or meaningful for the
9 majority of commands in this application. The middleware
10 requires a timeout to be supplied only where it is
11 meaningful to do so. WOSA also allows cancel commands to
12 be sent after any other command. Not all ATM functions
13 can really be cancelled and the middleware only provides
14 cancel commands where cancellation can actually be
15 achieved. The request IDs returned by WOSA for each
16 asynchronous operation are abstracted out by the
17 middleware. WOSA is accessed only by the middleware and
18 not directly by the application.

19
20 Clearly the preferred embodiment described above may
21 readily be adapted to operate with any operating system
22 or component system.

23
24 Further modifications and improvements may be
25 incorporated without departing from the scope of the
26 invention herein intended.

1 CLAIMS

2
3 1. A method for providing transaction services wherein

4
5 (a) the user of the transaction services interacts
6 with a computer-based transaction machine which is
7 controlled by one or more software applications;

8
9 (b) the software applications interact with the
10 functional interfaces of middleware software, which
11 extends the functionality of an underlying operating
12 system; and

13
14 (c) said functional interfaces provide functionality
15 which is implemented in a manner adapted to the
16 particular hardware capabilities of the transaction
17 machine.

18
19 2. A method for providing transaction services
20 according to Claim 1 wherein the transaction machine
21 is selected from a group which comprises automatic
22 teller machines, kiosks and electronic point of sale
23 machines.

24
25 3. A method for providing transaction services
26 according to any preceding Claim wherein middleware
27 software comprises a series of transaction objects
28 and controls for standard device functions.

29
30 4. A method for providing transaction services
31 according to Claim 3 wherein transaction objects are
32 independent of the interface between the user and

1 the transaction machine; the interface between the
2 user and the transaction machine being customisable.

3

4 5. A method for providing transaction services
5 according to Claim 3 or Claim 4 wherein controls
6 implement a capabilities interface.

7

8 6. A method for providing transaction services
9 according to Claim 5 wherein the capabilities
10 interface can communicate the capabilities of the
11 control software.

12

13 7. A method for providing transaction services
14 according to any of Claims 3 to 6 wherein
15 applications, objects and controls are concurrent
16 and asynchronous.

17

18 8. A method for providing transaction services
19 according to any of Claims 3 to 7 wherein controls
20 have a mode in which events are queued up and
21 delivered to the application on demand.

22

23 9. A method for providing transaction services
24 according to any of Claims 3 to 8 wherein controls
25 are adapted to run on the transaction machine even
26 when supported hardware devices are not present.

27

28 10. A method for providing transaction services
29 according to any preceding Claim wherein the
30 middleware software uses one or more open standards
31 for interacting with different hardware systems.

32

- 1 11. A method for providing transaction services
2 according to any preceding Claims wherein middleware
3 software only provides cancellation commands for
4 functions which can be successfully cancelled.
5
- 6 12. A method for providing transaction services
7 according to any preceding Claim wherein middleware
8 software only requires a timeout command to be
9 supplied when it is meaningful to do so.
10
- 11 13. A method for providing transaction services
12 according to any of Claims 3 to 12 wherein all
13 controls are persistent.
14
- 15 14. A method for providing transaction services
16 according to any of Claims 3 to 13 wherein there is
17 provided a control containing a persistent object.
18
- 19 15. A method for providing transaction services
20 according to any preceding Claim wherein all errors
21 and transgressions are asserted by the middleware
22 software.
23
- 24 16. A method for providing transaction services
25 according to any preceding Claim in which the
26 middleware software provides a trace facility that
27 is always enabled and which logs trace events.
28
- 29 17. A method for providing transaction services
30 according to Claim 16 wherein the middleware
31 software uses a ring buffer to store a log of trace
32 events.

1

2 18. A method for providing transaction services
3 according to Claim 17 wherein the middleware
4 software writes trace data to memory and then copies
5 it to disk only when the transaction machine is
6 idle.

7

8 19. A method for providing transaction services
9 according to any preceding Claim wherein one or more
10 software applications are hosted in a web browser.

11

12 20. A method for providing transaction services
13 according to Claim 19 wherein the use of a web
14 browser provides support for software distribution
15 and network connections.

16

17 21. A method for providing transaction services
18 according to Claim 19 or Claim 20 wherein an
19 additional browser frame is provided which contains
20 the device controls required to detect events which
21 must be dealt with immediately they occur.

22

23 22. A method for providing transaction services
24 according to any preceding Claim wherein middleware
25 software comprises a series of COM components with a
26 scriptable ActiveX® interface.

27

28 23. A method for providing transaction services
29 according to any preceding Claim wherein middleware
30 software comprises a series of Javabeans™ components
31 with a scriptable interface.

32

1 24. A method for providing transaction services
2 according to any of Claims 19 to 23 wherein use of a
3 web browser allows conventional web sites to be
4 displayed by the computer-based transaction machine.
5

6 25. A method for providing transaction services
7 according to Claim 24 wherein middleware software
8 allows or disallows access to particular web sites
9 according to a rule database.
10

11 26. A method for providing transaction services
12 according to Claim 24 or Claim 25 wherein middleware
13 software is adapted to customise time-out of the
14 display of individual internet web sites.
15

16 27. A method for providing transaction services
17 according to any preceding Claim wherein the
18 computer-based transaction machine is adapted to
19 allow the software applications and middleware to be
20 altered across a network by an authority.
21

22 28. A method for providing transaction services
23 according to any preceding Claim wherein the
24 transaction machine can communicate information
25 about their status to a remote monitoring station
26 across a network.
27

28 29. A computer-based transaction machine; wherein said
29 computer-based transaction machine is provided with
30 hardware devices for interaction with users and the
31 exchange of transaction-related information with
32 other machines; wherein said computer-based

1 transaction machine is controlled by one or more
2 software applications; wherein said software
3 applications control hardware devices through
4 functional interfaces with middleware software;
5 wherein said middleware software extends the
6 functionality of an underlying operating system and
7 wherein said functional interfaces are hardware
8 independent but provide functionality which is
9 implemented in a manner adapted to the capabilities
10 of the particular hardware devices which are
11 provided.

12
13 30. A computer-based transaction machine according to
14 Claim 29 wherein the transaction machine is selected
15 from a group which comprises automatic teller
16 machines, kiosks and electronic point of sale
17 machines.

18
19 31. A computer-based transaction machine according to
20 Claim 29 or Claim 30 wherein middleware software
21 comprises a series of transaction objects and
22 controls for standard device functions.

23
24 32. A computer-based transaction machine according to
25 Claim 31 wherein transaction objects are independent
26 of the interface between the user and the
27 transaction machine; the interface between the user
28 and the transaction machine being customisable.

29
30 33. A computer-based transaction machine according to
31 Claim 31 or Claim 32 wherein controls implement a
32 capabilities interface.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

34. A computer-based transaction machine according to Claim 33 wherein the capabilities interface can communicate the capabilities of the control software.
35. A computer-based transaction machine according to any of Claims 31 to 34 wherein applications, objects and controls are concurrent and asynchronous.
36. A computer-based transaction machine according to any of Claims 31 to 35 wherein controls have a mode in which events are queued up and delivered to the application on demand.
37. A computer-based transaction machine according to any of Claims 31 to 36 wherein controls are adapted to run on the transaction machine even when supported hardware devices are not present.
38. A computer-based transaction machine according to any of Claims 29 to 37 wherein the middleware software uses one or more open standards for interacting with different hardware systems.
39. A computer-based transaction machine according to any of Claims 29 to 38 wherein middleware software only provides cancellation commands for functions which can be successfully cancelled.
40. A computer-based transaction machine according to any of Claims 29 to 39 wherein middleware software

1 only requires a timeout command to be supplied when
2 it is meaningful to do so.

3

4 41. A computer-based transaction machine according to
5 any of Claims 31 to 40 wherein all controls are
6 persistent.

7

8 42. A computer-based transaction machine according to
9 any of Claims 31 to 41 wherein there is provided a
10 control containing a persistent object.

11

12 43. A computer-based transaction machine according to
13 any of Claims 29 to 42 wherein all errors and
14 transgressions are asserted by the middleware
15 software.

16

17 44. A computer-based transaction machine according to
18 any of Claims 29 to 43 wherein the middleware
19 software provides a trace facility that is always
20 enabled and which logs trace events.

21

22 45. A computer-based transaction machine according to
23 Claim 44 wherein the middleware software uses a ring
24 buffer to store a log of trace events.

25

26 46. A computer-based transaction machine according to
27 Claim 45 wherein the middleware software writes
28 trace data to memory and then copies it to disk only
29 when the transaction machine is idle.

30

- 1 47. A computer-based transaction machine according to
2 any of Claims 29 to 46 wherein one or more software
3 applications are hosted in a web browser.
4
- 5 48. A computer-based transaction machine according to
6 Claim 47 wherein the use of a web browser provides
7 support for software distribution and network
8 connections.
9
- 10 49. A computer-based transaction machine according to
11 Claim 47 or Claim 48 wherein an additional browser
12 frame is provided which contains the device controls
13 required to detect events which must be dealt with
14 immediately they occur.
15
- 16 50. A computer-based transaction machine according to
17 any of Claims 29 to 49 wherein middleware software
18 comprises a series of COM components with a
19 scriptable ActiveX* interface.
20
- 21 51. A computer-based transaction machine according to
22 any of Claims 29 to 50 wherein middleware software
23 comprises a series of Javabeans™ components with a
24 scriptable interface.
25
- 26 52. A computer-based transaction machine according to
27 any of Claims 47 to 51 wherein use of a web browser
28 allows conventional web sites to be displayed by the
29 computer-based transaction machine.
30
- 31 53. A computer-based transaction machine according to
32 Claim 52 wherein middleware software allows or

1 disallows access to particular web sites according
2 to a rule database.

3

4 54. A computer-based transaction machine according to
5 Claim 52 or Claim 53 wherein middleware software is
6 adapted to customise time-out of the display of
7 individual internet web sites.

8

9 55. A computer-based transaction machine according to
10 any of Claims 29 to 54 wherein the computer-based
11 transaction machine is adapted to allow the software
12 applications and middleware to be altered across a
13 network by an authority.

14

15 56. A computer-based transaction machine according to
16 any of Claims 29 to 55 wherein the transaction
17 machine can communicate information about their
18 status to a remote monitoring station across a
19 network.

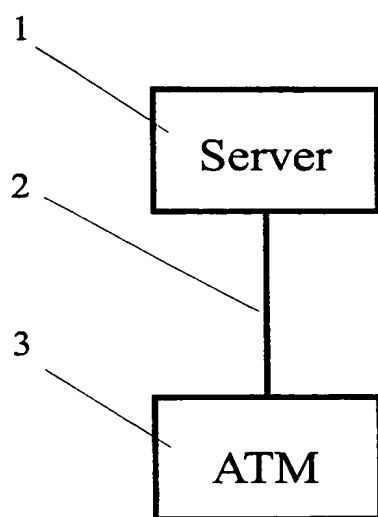
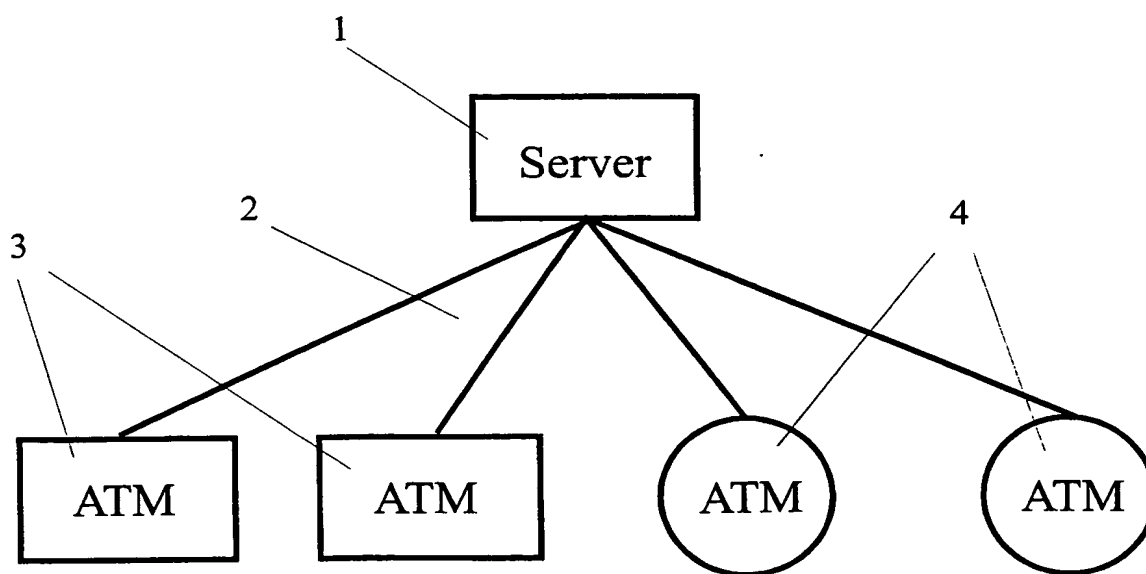
20

21 57. A network comprising a plurality of computer-based
22 transaction machines according to any of Claims 29
23 to 56, one or more networking means and one or more
24 application servers.

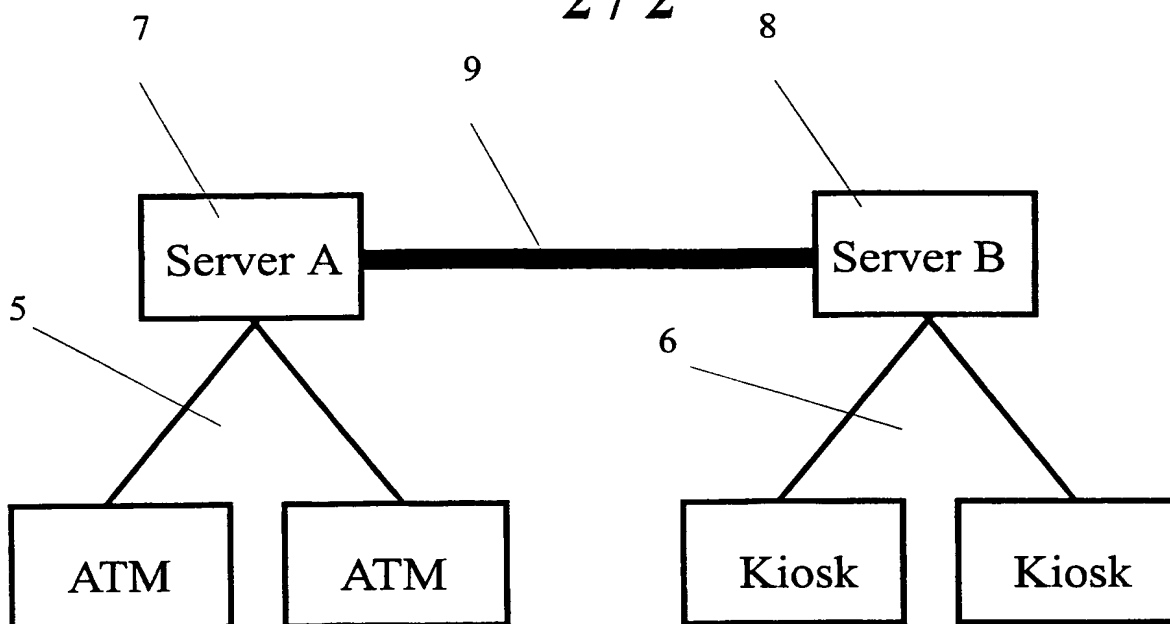
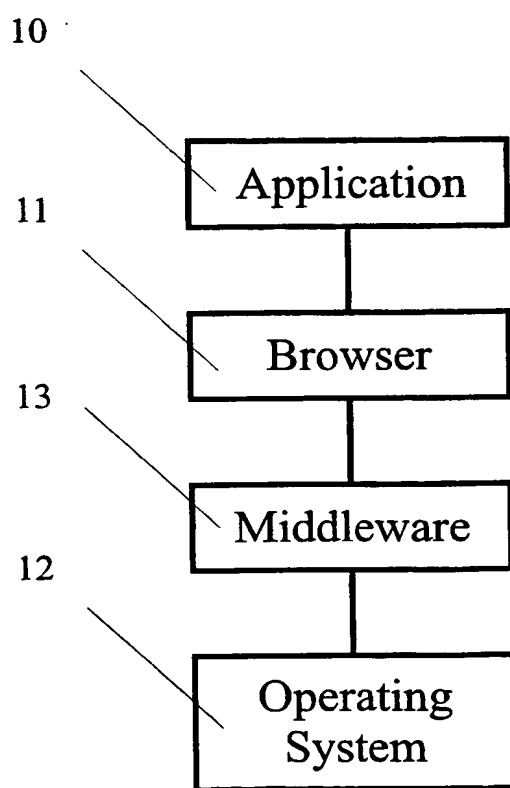
25

26 58. An Extranet formed by combining a plurality of
27 networks of computer-based transaction machines
28 according to Claim 57.

- 1 59. An Extranet according to Claim 58 provided with a
- 2 security mechanism which limits the hardware
- 3 functionality available to individual software
- 4 applications.

1 / 2**Figure 1****Figure 2**

2 / 2

**Figure 3****Figure 4**